```
--File: WEBreak.mesa
--Edited by:
--          Johnsson  July 22, 1978  1:16 PM
--          Sandman  May 4, 1978  10:56 AM
--          Barbara  June 26, 1978  10:25 AM

DIRECTORY
  AltoDefs: FROM "altodefs" USING [CharsPerPage],
  AltoFileDefs: FROM "altofiledefs" USING [FA],
  ControlDefs: FROM "controldefs" USING [BytePC, GlobalFrameHandle],
  DebugBreakptDefs: FROM "debugbreakptdefs" USING [
    BreakPointError, BTtype, EXOItype, InsertBreak, RemoveBreak, SCtype],
  DebugContextDefs: FROM "debugcontextdefs" USING [
    FrameToModuleName, ModuleNameToFrame, MultipleInstances],
  DebugData: FROM "debugdata" USING [gContext],
  DebuggerDefs: FROM "debuggerdefs" USING [PcToBTI],
  DebugMiscDefs: FROM "debugmiscdefs" USING [LookupFail],
  DebugSymbolDefs: FROM "debugsymboldefs" USING [
    DAcquireSymbolTable, DReleaseSymbolTable, SymbolsForGFrame],
  DebugUtilityDefs: FROM "debugutilitydefs" USING [LoadStateInvalid],
  IODefs: FROM "iodefs" USING [ControlZ, CR, SP],
  LoadStateDefs: FROM "loadstatedefs" USING [
    InputLoadState, ReleaseLoadState],
  KeyDefs: FROM "keydefs" USING [Keys],
  MenuDefs: FROM "menudefs" USING [
    ClearMenu, CreateMenu, DisplayMenu, MarkMenuItem, MenuItem],
  RectangleDefs: FROM "rectangledefs" USING [
    CursorToMapCoords, InvertBoxInRectangle, Rptr],
  StreamDefs: FROM "streamdefs" USING [
    EqualIndex, GetFA, JumpToFA, NormalizeIndex, StreamError],
  StringDefs: FROM "stringdefs" USING [AppendChar, AppendString, EqualString],
  SymbolTableDefs: FROM "symboltabledefs" USING [
    NoSymbolTable, SymbolTableBase],
  SymDefs: FROM "symdefs" USING [FGTEntry],
  WindExDefs: FROM "windexdefs" USING [
    CursorToRectangleCoords,GetMouseButton, NullIndex, SetCursor,
    WEDataHandle, xcursorloc, ycursorloc],
  WindowDefs: FROM "windowdefs" USING [
    DiskHandle, GetCurrentDisplayWindow, GetSelection, MarkSelection,
    StreamIndex, UpdateSelection, WindowHandle, xCoord, yCoord];

DEFINITIONS FROM StreamDefs, WindowDefs, WindExDefs;

WEBreak: PROGRAM [WEState: WEDataHandle]
  IMPORTS DebugBreakptDefs, DebugContextDefs, DebuggerDefs, DebugMiscDefs,
    DebugSymbolDefs, DebugUtilityDefs, LoadStateDefs, MenuDefs, RectangleDefs,
    StreamDefs, StringDefs, SymbolTableDefs, WindExDefs, WindowDefs,
    DDptr: DebugData
  EXPORTS WindExDefs
  SHARES StreamDefs, WindExDefs =

BEGIN

OPEN WEState;

CR: CHARACTER = 15C;

-- some externals

nCommands: CARDINAL = 12;

create: CARDINAL = 0;
destroy: CARDINAL = 1;
move: CARDINAL = 2;
grow: CARDINAL = 3;
load: CARDINAL = 4;
stuff: CARDINAL = 5;
find: CARDINAL = 6;
break: CARDINAL = 7;
clear: CARDINAL = 8;
trace: CARDINAL = 9;
position: CARDINAL = 10;
keys: CARDINAL = 11;

MenuSelect: PUBLIC PROCEDURE [w: WindowHandle, x: xCoord, y: yCoord]=
  BEGIN OPEN MenuDefs;
```

```
  -- define locals
  index: INTEGER ← -1;
  mapx: xCoord;
  mapy: yCoord;
  defaultmenu:  DESCRIPTOR FOR ARRAY OF MenuItem =
    DESCRIPTOR[BASE[menuarray], LENGTH[menuarray]];
  -- check if a menu
  IF w.menu = NIL THEN w.menu ← CreateMenu[defaultmenu];
  -- paste it up there
  [mapx, mapy] ← RectangleDefs.CursorToMapCoords[defaultmapdata, x, y];
  mapy ← MIN[mapy, MAX[0,(w.rectangle.bitmap.height)
     -(LENGTH[w.menu.array]+LOOPHOLE[defaultlineheight,INTEGER]+2)]];
  SetCursor[menu];
  DisplayMenu[w.menu, w.rectangle.bitmap, mapx, mapy];
  -- while the button is down select menu items
  WHILE GetMouseButton[] = Blue DO
    -- convert to rectangle coords
    x ← xcursorloc↑;
    y ← ycursorloc↑;
    -- and see if in menu
    [x, y] ← CursorToRectangleCoords[w.menu.rectangle, x, y];
    index ← IF x > 0 AND x <= w.menu.rectangle.cw AND y > 0 AND
      y <= w.menu.rectangle.ch THEN y/defaultlineheight ELSE -1;
    IF index >= LENGTH[w.menu.array] THEN index ← -1;
    MarkMenuItem[w.menu, index];
  ENDLOOP;
  -- and restore menus region and contents underneath
  ClearMenu[w.menu];
  -- see if command selected
  IF index # -1 THEN w.menu.array[index].proc[w, xcursorloc↑, ycursorloc↑];
  END;


FGFrame: TYPE = RECORD [
  fge: SymDefs.FGTEntry,
  frame: ControlDefs.GlobalFrameHandle];

SetBreak: PROCEDURE [w: WindowHandle, x: xCoord, y: yCoord] =
  BEGIN
  cond: STRING = IF KeyDefs.Keys.Ctrl = down THEN "1"L ELSE NIL;
  Breakpt[w, set, break, cond];
  END;

Clear: PROCEDURE [w: WindowHandle, x: xCoord, y: yCoord] =
  BEGIN
  Breakpt[w, clear, break, NIL];
  END;

SetTrace: PROCEDURE [w: WindowHandle, x: xCoord, y: yCoord] =
  BEGIN
  Breakpt[w, set, trace, NIL];
  END;

SCtype: TYPE = DebugBreakptDefs.SCtype;
BTtype: TYPE = DebugBreakptDefs.BTtype;

FlashWindow: PUBLIC PROCEDURE [w: WindowHandle] =
  BEGIN OPEN RectangleDefs;
  r: Rptr = w.rectangle;
  InvertBoxInRectangle[r, 0, r.cw, 0, r.ch];
  THROUGH [0..5000) DO NULL ENDLOOP;
  InvertBoxInRectangle[r, 0, r.cw, 0, r.ch];
  END;

Breakpt: PROCEDURE [w: WindowHandle, sc: SCtype, bt: BTtype, cond: STRING] =
  BEGIN OPEN StringDefs, DebugBreakptDefs;
  str: STRING;
  fgf: FGFrame;
  IF w.type # file THEN RETURN;
  str ← GetSelection[w];
  IF EqualIndex[w.selection.rightindex, NullIndex] THEN RETURN;
  SetCursor[hourglass];
  BEGIN
    ENABLE SymbolTableDefs.NoSymbolTable, BreakPointError,
      DebugMiscDefs.LookupFail, DebugContextDefs.MultipleInstances,
      DebugUtilityDefs.LoadStateInvalid => BEGIN FlashWindow[w]; CONTINUE END;
```

```
  SELECT TRUE FROM
    EqualString[str, "PROCEDURE"L] =>
      BEGIN
      fgf ← SetupBreakPoint[w, w.selection.leftindex, entry];
      IF sc = set THEN InsertBreak[fgf.frame,[fgf.fge.cindex],entry,bt,cond]
      ELSE RemoveBreak[fgf.frame, [fgf.fge.cindex]];
      IF GetCurrentDisplayWindow[] = w THEN
        BEGIN MarkSelection[w]; UpdateSelection[w]; END;
      END;
    EqualString[str,"RETURN"L] =>
      BEGIN
      fgf ← SetupBreakPoint[w, w.selection.leftindex, exit];
      IF sc = set THEN InsertBreak[fgf.frame,[fgf.fge.cindex],exit,bt,cond]
      ELSE RemoveBreak[fgf.frame, [fgf.fge.cindex]];
      IF GetCurrentDisplayWindow[] = w THEN
        BEGIN MarkSelection[w]; UpdateSelection[w]; END;
      END;
    ENDCASE =>
      BEGIN
      fgf ← SetupBreakPoint[w, w.selection.leftindex, in];
      IF sc = set THEN InsertBreak[fgf.frame,[fgf.fge.cindex],in,bt,cond]
      ELSE RemoveBreak[fgf.frame, [fgf.fge.cindex]];
      IF GetCurrentDisplayWindow[] = w THEN
        BEGIN
        MarkSelection[w];
        w.selection.leftindex ← w.selection.rightindex ←
          NormalizeIndex[[0,fgf.fge.findex]];
        UpdateSelection[w];
        END;
      END;
  END;
  ButtonWait[];
  SetCursor[textpointer];
  RETURN;
  END;

SetupBreakPoint: PROCEDURE [
  w: WindowHandle, index: StreamIndex, type: DebugBreakptDefs.EXOItype]
  RETURNS [fgf: FGFrame] =
  BEGIN OPEN SymbolTableDefs, DebugSymbolDefs;
  symbase: SymbolTableBase;
  delta: CARDINAL ← 10000;
  i,x,besti: CARDINAL ← 0;
  module: STRING ← [40];
  indexx: CARDINAL ← AltoDefs.CharsPerPage*index.page + index.byte;
  IF ~LittleParser[w, module]
    THEN SIGNAL DebugMiscDefs.LookupFail["PROGRAM"L];
  fgf.frame ← GetContext[module];
  symbase ← DAcquireSymbolTable[SymbolsForGFrame[fgf.frame]];
  BEGIN OPEN symbase;
  FOR i IN [0..LENGTH[fgTable]) DO
    IF indexx >= (x ← fgTable[i].findex) AND (x ← indexx-x) < delta THEN
      BEGIN
      delta ← x; besti ← i;
      IF delta = 0 THEN EXIT;
      END;
    ENDLOOP;
  SELECT type FROM
    entry =>
      BEGIN
      fgf.fge.cindex ← fgTable[besti+1].cindex;
      fgf.fge.findex ← fgTable[besti+1].findex;
      END;
    exit =>
      WITH (bb+DebuggerDefs.PcToBTI[symbase, [fgTable[besti].cindex]]).info
        SELECT FROM
          External => BEGIN
            fgf.fge.cindex ← ControlDefs.BytePC[origin+bytes-1];
            fgf.fge.findex ← fgTable[startIndex+indexLength-1].findex;
            END;
          ENDCASE;
    in =>
      BEGIN
      fgf.fge.cindex ← fgTable[besti].cindex;
      fgf.fge.findex ← fgTable[besti].findex;
      END;
```

```
    ENDCASE;
  DReleaseSymbolTable[symbase];
  END;
  RETURN
  END;

GetContext: PROCEDURE [module: STRING]
  RETURNS [frame: ControlDefs.GlobalFrameHandle] =
  BEGIN
  currentContext: STRING ← [40];
  [] ← LoadStateDefs.InputLoadState[];
  DebugContextDefs.FrameToModuleName[DDptr.gContext, currentContext !
    UNWIND => LoadStateDefs.ReleaseLoadState[]];
  LoadStateDefs.ReleaseLoadState[];
  frame ← IF StringDefs.EqualString[module, currentContext] THEN
    DDptr.gContext ELSE DebugContextDefs.ModuleNameToFrame[module];
  RETURN
  END;

LittleParser: PROCEDURE [w: WindowHandle, module: STRING]
  RETURNS[BOOLEAN] =
  BEGIN
  char: CHARACTER;
  testId: STRING ← [40];
  fa: AltoFileDefs.FA;
  found: BOOLEAN ← FALSE;
  module.length ← 0;
  GetFA[w.file, @fa];
  w.file.reset[w.file];
  WHILE ~found DO
    ENABLE StreamError, NotFound => BEGIN module.length ← 0; EXIT END;
    GetId[w.file,testId];
    SELECT TRUE FROM
      StringDefs.EqualString[testId, "DIRECTORY"L],
      StringDefs.EqualString[testId, "DEFINITIONS"L] =>
        UNTIL (char ← w.file.get[w.file]) = '; DO NULL; ENDLOOP;
      ENDCASE =>
        BEGIN OPEN StringDefs;
        AppendString[module,testId];
        GetId[w.file,testId];
        IF EqualString[testId, "PROGRAM"L] OR EqualString[testId, "MONITOR"L]
          THEN found ← TRUE;
        EXIT;
        END;
    ENDLOOP;
  JumpToFA[w.file, @fa];
  RETURN[found]
  END;

NotFound: SIGNAL = CODE;

GetId: PROCEDURE[file: DiskHandle, token: STRING] =
  BEGIN OPEN IODefs;
  c: CHARACTER ← file.get[file];
  token.length ← 0;
  DO
    SELECT c FROM
      '- =>
        IF (c ← file.get[file]) = '- THEN
          DO
            SELECT file.get[file] FROM
              CR => BEGIN c ← CR; EXIT END;
              '- => IF file.get[file] = '- THEN
                BEGIN c ← file.get[file]; EXIT END;
              ENDCASE;
            ENDLOOP
        ELSE SIGNAL NotFound;
      ControlZ => UNTIL file.get[file] = CR DO
        NULL REPEAT FINISHED => c ← CR ENDLOOP;
      SP, CR => IF token.length # 0 THEN RETURN ELSE c ← file.get[file];
      '=, '[ => SIGNAL NotFound;
      ': => RETURN;
      < 40C => c ← file.get[file];
      ENDCASE =>
        BEGIN StringDefs.AppendChar[token, c]; c ← file.get[file]; END;
    ENDLOOP;
```

```
  RETURN;
  END;

ButtonWait: PROCEDURE =
  BEGIN
  -- wait until all button are up
  UNTIL GetMouseButton[] = None DO NULL; ENDLOOP;
  RETURN;
  END;

-- initialization for windows module

InitBreak: PROCEDURE =
  BEGIN OPEN MenuDefs;
  menuarray[break] ← MenuItem["Set Brk", SetBreak];
  menuarray[clear] ← MenuItem["Clr Brk", Clear];
  menuarray[trace] ← MenuItem["Set Trc", SetTrace];
  END;


InitBreak[];

END. of WEBreak
```